Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork[⋆]

Peter Stone and Manuela Veloso

Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213, USA

{pstone,veloso}@cs.cmu.edu

Abstract. Multi-agent domains consisting of teams of agents that need to collaborate in an adversarial environment offer challenging research opportunities. In this paper, we introduce *periodic team synchronization* domains, as time-critical environments in which agents act autonomously with limited communication, but they can periodically synchronize in a full-communication setting. We present a team agent structure that allows for an agent to capture and reason about team agreements. We achieve collaboration between agents through the introduction of *formations*. A formation decomposes the task space defining a set of *roles*. Homogeneous agents can flexibly switch roles within formations, and agents can change formations dynamically, according to pre-defined triggers to be evaluated at run-time. This flexibility increases the performance of the overall team. Our team structure further includes pre-planning for frequent situations. We fully implemented this approach in the domain of robotic soccer. Our simulator team made it to the semi-finals of the RoboCup-97 competition, in which 29 teams participated. It achieved a total score of 67–9 over six different games, and successfully demonstrated its flexible team structure. Using the same team structure, our small robot team won the RoboCup-97 small-robot competition, in which 4 teams participated. It achieved a total score of 13–1 over 4 games and also demonstrated its flexible team structure.

1 Introduction

A multi-agent system which involves several agents that collaborate towards the achievement of joint objectives is viewed as a *team* of agents. Most proposed teamwork structures (e.g. joint intentions, shared plans) rely on agents in a multi-agent system to negotiate and/or contract with each other in order to initiate team plans [4, 7, 8]. However, in dynamic, real-time domains with unreliable communication, complex negotiation protocols may take too much time and/or be infeasible due to communication restrictions.

Our work has been focused in time-critical environments in which agents in a team alternate between periods of limited and unlimited communication. This focus

[⋆] This research is sponsored in part by the DARPA/RL Knowledge Based Planning and Scheduling Initiative under grant number F30602-97-2-0250. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U. S. Government.

leads us to introduce the concept of *Periodic Team Synchronization* (PTS) domains. In PTS domains, during the limited (or no) communication periods, agents need to act autonomously, while still working towards a common team goal. Time-critical environments require real-time response and therefore eliminate the possibility of heavy communication between team agents. However, in PTS domains, agents can periodically synchronize in a safe, full-communication setting. In this paper, we introduce a flexible teamwork structure that allows for task decomposition and dynamic role assignment in PTS domains.

In PTS domains, teams are long-term entities so that it makes sense for them to have periodic, reliable, private synchronization intervals in which they can form off-line agreements for future use in unreliable, time-critical environments. This view of teams is complementary to teams that form on the fly for a specific action and keep communicating throughout the execution of that action as in [4]. Instead, in PTS domains, teams define coordination protocols during the synchronization opportunity and then disperse into the environment, acting autonomously with little or no communication possible.

It has been claimed that pre-determined team actions are not flexible or robust to failure [26]. A key contribution of our work is the demonstration that pre-determined multi-agent protocols can facilitate effective teamwork while retaining flexibility in PTS domains. We call these pre-determined protocols *locker-room agreements*. Formed during the periodic synchronization opportunities, locker-room agreements are remembered identically by all agents and allow them to coordinate efficiently. In the context of [3], locker-room agreements can be viewed as C-commitments, or commitments by team members to do the appropriate thing at the right time, as opposed to S-commitments with which agents adopt each other's goals. In the context of [5], the creation of a locker-room agreement is norm acceptance while its use is norm compliance.

In this paper, we introduce an agent architecture suited for team agents in PTS domains. The architecture allows for an agent to act appropriately based on locker-room agreements. Within the framework presented in [15], the architecture is for interactive software and hardware multi-agents.

Since the agents act autonomously and sense the world individually, they may have different views of what is best for the team. However, contrary to self-interested agents for which coordination may or may not be rational [2, **?**], our agents have no individual incentives. Their performance is measured as a unit: each agent's highest goal is the success of the team.

A straightforward approach to PTS domains is to break the task at hand into multiple rigid roles, assigning one agent to each role. Thus each component of the task is accomplished and there are no conflicts among agents in terms of how they should accomplish the team goal. However such an approach is subject to several problems: inflexibility to short-term changes (e.g. one robot is non-operational), inflexibility to long-term changes (e.g. a route is blocked), and a lack of facility for reassigning roles.

We introduce instead *formations* as a team structure. A formation decomposes the task space defining a set of roles with associated behaviors. In a general scenario with heterogeneous agents, subsets of homogeneous agents can flexibly switch roles within formations, and agents can change formations dynamically. This flexibility increases the performance of the overall team. The homogeneous assumption underlying the desired flexible role-switching behavior creates a challenge in terms of determining if and when they should switch roles.

Within these PTS domains and our flexible teamwork structure, several challenges arise. For example, how to represent and follow locker-room agreements; how to determine the appropriate times for agents to change roles and/or formations; how to ensure that all agents are using the same formation; and how to ensure that all roles in a formation are filled. Since the agents are autonomous and do not share memory, they could easily become uncoordinated.

In a nutshell, the main contributions of this paper are: the introduction of the concepts of PTS domains and locker-room agreements; the definition of a general team agent architecture structure for defining a flexible teamwork structure; the facilitation of smooth transitions among roles and entire formations; and a method for using roles to define pre-compiled multi-step, multi-agent plans.

Our work is situated in an example of a PTS domain in which we conducted our research, robotic soccer [10]. In both simulated and robotic systems, teams can plan strategies before the game, at halftime, or at other breakpoints, but during the course of the game, communication is limited. There are several other examples of PTS domains, such as hospital/factory maintenance [6], multi-spacecraft missions [20], search and rescue, and battlefield combat [26].

2 Team Member Architecture

Our new teamwork structure is situated within a team member architecture suitable for PTS domains in which individual agents can capture locker-room agreements and respond to the environment, while acting autonomously. Based on a standard agent paradigm, our team member architecture allows agents to sense the environment, to reason about and select their actions, and to act in the real world. At team synchronization opportunities, the team also makes a locker-room agreement for use by all agents during periods of low communication. Figure 1 shows the functional input/output model of the architecture.

The agent keeps track of three different types of state: the *world state*, the *locker-room agreement*, and the *internal state*. The agent also has two different types of behaviors: *internal behaviors* and *external behaviors*.

The World State reflects the agent's conception of the real world, both via its sensors and via the predicted effects of its actions. It is updated as a result of processed sensory information. It may also be updated according to the predicted effects of the external behavior module's chosen actions. The world state is directly accessible to both internal and external behaviors.

The Locker-Room Agreement is set by the team when it is able to privately synchronize. It defines the flexible teamwork structure as presented below as well as inter-agent communication protocols. The locker-room agreement may change periodically when the team is able to re-synchronize; however, it generally remains unchanged. The locker-room agreement is accessible only to internal behaviors.

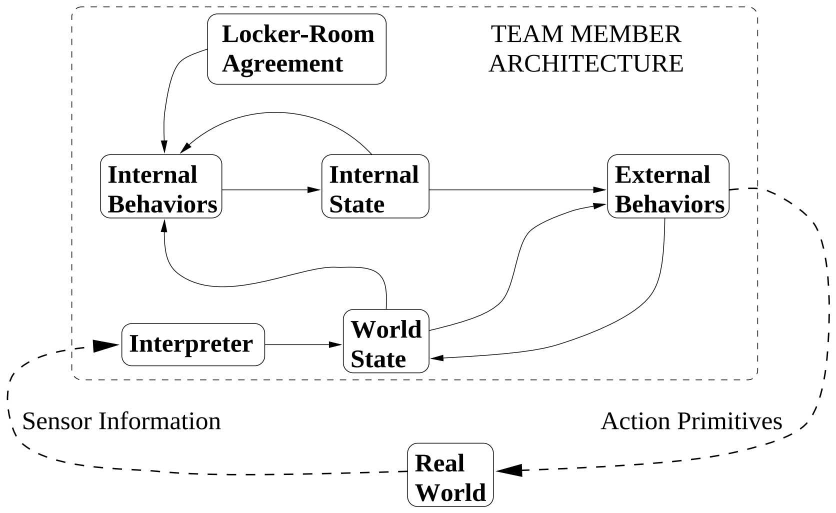

Fig. 1. The team member architecture for PTS domains.

The Internal State stores the agent's internal variables. It may reflect previous and current world states, possibly as specified by the locker-room agreement. For example, the agent's role within a team behavior could be stored as part of the internal state, as could a distribution of past world states. The agent updates its internal state via its internal behaviors.

The Internal Behaviors update the agent's internal state based on its current internal state, the world state, and the team's locker-room agreement.

The External Behaviors reference the world and internal states, sending commands to the actuators. The actions affect the real world, thus altering the agent's future percepts. External behaviors consider only the world and internal states, without direct access to the locker-room agreement.

Internal and external behaviors are similar in structure, as they are both sets of condition/action pairs where conditions are logical expressions over the inputs and actions are themselves behaviors as illustrated in Figure 2. In both cases, a behavior is a directed acyclic graph (DAG) of arbitrary depth. The leaves of the DAGs are the behavior types' respective outputs: internal state changes for internal behaviors and action primitives for external behaviors.

Our notion of behavior is consistent with that laid out in [13]. In particular, behaviors can be nested at different levels: selection among lower-level behaviors can be considered a higher-level behavior, with the overall agent behavior considered a single "do-the-task" behavior. There is one such *top-level* internal behavior and one top-level external behavior; they are called when it is time to update the internal state or act in the world, respectively. We now introduce the team structure that builds upon this team member architecture.

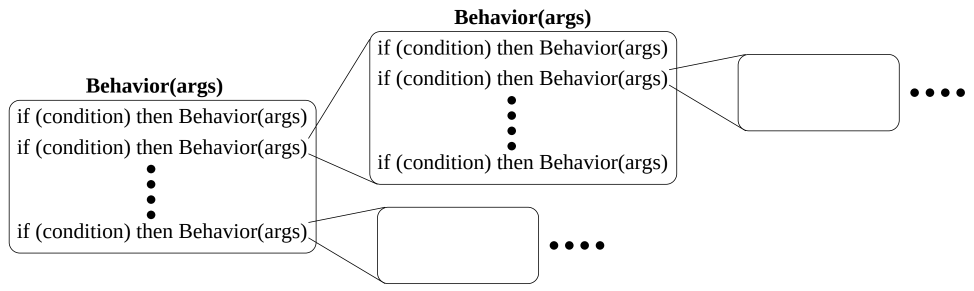
Behavior(args)
if (condition) then Behavior(args)
if (condition) then Behavior(args)
if (condition) then Behavior(args)
Behavior(args)
if (condition) then Behavior(args)
if (condition) then Behavior(args)
if (condition) then Behavior(args)

Fig. 2. Internal and external behaviors are organized in a directed acyclic graph.

3 Team Structure

Common to all players, the locker-room agreement defines the team structure while team members are acting in a time-critical environment with low bandwidth communication. In this section, we introduce a structure for capturing locker-room agreements. It defines sets of agent roles with protocols for switching among them. It can also define multi-step multi-agent plans for execution in specific situations. It indirectly affects the agent external behaviors by changing the agents' internal states via internal behaviors.

Our teamwork structure involves *flexible roles* that are organized into *formations*, which we now introduce.

3.1 Role

A *role*, r, consists of a specification of an agent's internal and external behaviors. The conditions and arguments of any behavior can depend on the agent's current role, which is a function of its internal state. At the extreme, a top-level behavior could be a switch, calling an entirely different behavior graph for each possible role. However, the role can affect the agent's overall behavior at any level of its complete behavior graph. Notice that roles need not be rigid: by specifying ranges of parameters or behavior options, the agent filling role r can be given an arbitrary amount of flexibility.

For example, a role in the robotic soccer domain, can be a position such as a midfielder. In the hospital maintenance domain, a role could specify the wing of the hospital whose floors the appropriate agent should keep clean, while in the web search domain, it could specify a server to search.

3.2 Formation

We achieve collaboration between agents through the introduction of *formations* as a team structure. A formation decomposes the task space defining a set of roles. Formations include as many roles as there are agents in the team, so that each role is filled by one agent. In addition, formations can specify sub-formations, or *units*, that do not involve the whole team. A unit consists of a subset of roles from the formation, a *captain*, and intra-unit interactions among the roles.

For a team of n agents $A = \{a_1, a_2, \ldots, a_n\}$, any formation is of the form $F = \{R, \{U_1, U_2, \ldots, U_k\}\}$ where R is a set of roles $R = \{r_1, r_2, \ldots, r_n\}$ such that $i \neq j \Rightarrow r_i \neq r_j$. Note that there are the same number of roles as there are agents. Each unit U_i is a subset of R: $U_i = \{r_{i1}, r_{i2}, \ldots, r_{ik}\}$ such that $r_{ia} \in R$, $a \neq b \Rightarrow r_{ia} \neq r_{ib}$ and r_{i1} is the captain. The map $A \mapsto R$ is not pre-specified: roles can be filled by different homogeneous agents. A single role may be a part of any number of units and formations.

Formations can affect the agent's external behaviors by specifying inter-role interactions. Since roles can be re-used among formations, their formation-specific interactions cannot be included in the role definitions. Instead these interactions are part of the formation specification.

Units are used to deal with local problem solving issues. Rather than involving the entire team in a sub-problem, the roles that address it are organized into a unit.

Roles and formations are introduced independently from the agents that are to fill them. The locker-room agreement specifies an initial formation, a map from agents to roles, and run-time triggers for dynamic changing of formations. At any given time, each agent should know what formation the team is currently using. Agents keep mappings $A \mapsto R$ from teammates to roles in the current formation. All this team structuring information is stored in the agent's internal state. It can be altered via the agent's internal behaviors. Thus, in all, the locker-room agreement is used to coordinate task decomposition among agents, to coordinate dynamic team re-alignment during time-critical stages, and for defining pre-compiled multi-agent plans. The locker-room agreement can be hard-wired or it can be the result of automatic deliberative multi-agent planning. Figure 3 illustrates a team of agents smoothly switching roles and formations over time.

Since agents are autonomous and operating in a PTS domain, during the periods of limited communication there is no guarantee that they will all think that the team is using the same formation, nor that they have accurate maps $A \mapsto R$. In fact, the only guarantee is that each agent knows its own current role. Efficient low-bandwidth communication protocols allow agents to inform each other of their roles periodically. Further details on our implemented low-bandwidth communication protocol can be found in [22].

Similarly, communication can be used as an alternative to changing formations using run-time triggers, or as a back-up should an agent not observe a run-time trigger. Although communication can be useful, we create robust behaviors for team agents which ensure that the behaviors never absolutely depend upon having correct, up-to-date knowledge of teammates' internal states: they must degrade gracefully.

4 Implementation in Robotic Soccer

Robotic soccer is a very good example of a PTS domain: teams can coordinate before the game, at half-time, and at other break points, but communication is limited during play [10, 12]. Robotic soccer systems have been recently developed both in simulation [14, 23, 24] and with real robots [1, 9, 18, 19, 27]. The research presented in this paper was first developed in simulation and it has also been successfully used on our real robot team.

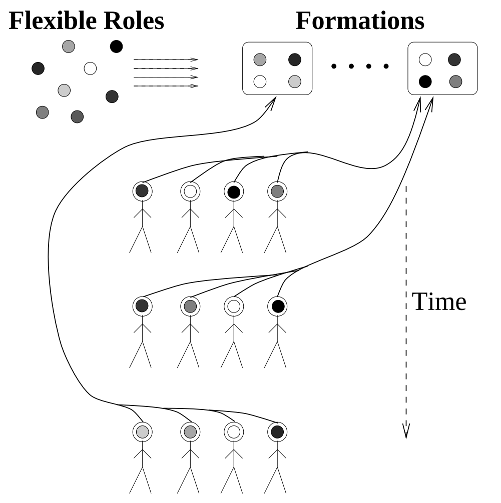

Fig. 3. Flexible roles and formations. Different roles are represented as differently shaded circles. Formations are possibly overlapping collections of roles. All roles and formations are known to all players. A player's current role is indicated by the shaded circle in its head and its current formation is indicated by an arrow to the formation. The players first switch roles while staying in the same formation; then they switch to an entirely new formation.

The soccer server [16], version 3 of which serves as the substrate simulator for the research reported in this paper, captures enough real-world complexities to be a very challenging domain. This simulator is realistic in many ways: (i) the players' vision is limited; (ii) the players can communicate by posting to a blackboard that is visible (but not necessarily intelligible) to all players; (iii) each player is controlled by a separate process; (iv) each team has 11 members; (v) players have limited stamina; (vi) actuators and sensors are noisy; (vii) dynamics and kinematics are modelled; and (viii) play occurs in *real time*: the agents must react to their sensory inputs at roughly the same speed as human or robotic soccer players. The Soccer Server was successfully used as the basis for the RoboCup-97 simulator competition in which 29 teams participated [10].

One approach to task decomposition in the Soccer Server is to assign fixed positions to agents.[2] Such an approach leads to several problems: i) short-term inflexibility in that the players cannot adapt their positions to the ball's location on the field; ii) long-term inflexibility in that the team cannot adapt to opponent strategy; and iii) local inefficiency in that players often get tired running across the field back to their positions after chasing the ball. Our formations allow for flexible teamwork and combat these problems. (As the term "position" is often used to denote the concept of "role" in the soccer domain, in this section we use the two terms interchangeably.)

[2] One of the teams in Pre-RoboCup-97 (IROS'96) used and depended upon these assignments: the players would pass to the fixed positions regardless of whether there was a player there.

4.1 Domain Instantiations of Roles and Formations

Figure 4 shows a sample top-level external behavior used by a team agent. The agent's top priority is to locate the ball. If the ball's location is known, it moves towards the ball or goes to its position (i.e., to assume its role), depending on its internal state. It also responds to any requested communications from teammates.

External Behavior: Play Soccer()

If (Ball Lost)	Find Ball()
If (Ball known AND Chasing)	Handle Ball(args1)
If (Ball known AND Not Chasing)	Play Position(args2)
If (Commuincate Flag Set)	Communicate()

Fig. 4. An example of a top-level external behavior for a robotic soccer player.

The referenced "Handle Ball" and "Play Position" behaviors may be affected by the agent's current role and/or formation. Such effects are realized by references to the internal state either at the level of function arguments (args1, args2), or within sub-behaviors. None of the actions in the condition-action pairs here are action primitives; rather, they are calls to lower level behaviors.

The definition of a position includes *home coordinates*, a *home range*, and a *maximum range*, as illustrated in Figure 5. The position's home coordinates are the default location to which the agent should go. However, the agent has some flexibility, being able to set its actual home position anywhere within the home range. When moving outside of the max range, the agent is no longer considered to be in the position. The home and max ranges of different positions can overlap, even if they are part of the same formations.

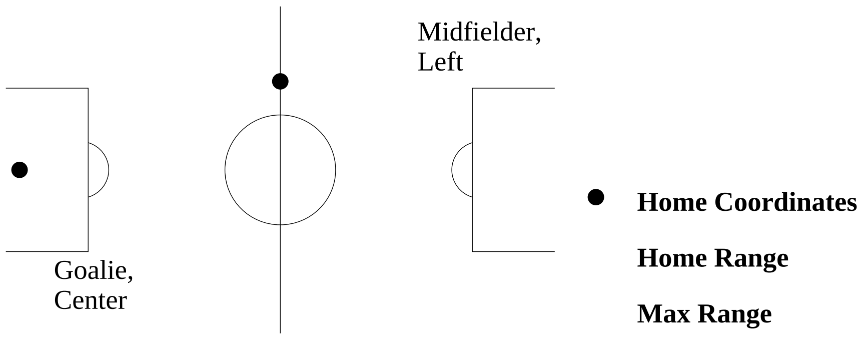

Fig. 5. Different positions with home coordinates and home and max ranges.

A formation consists of a set of positions and a set of units (as defined in Section 3.2). The formation and each of the units can also specify inter-position behavior specifications for the member positions, as illustrated in Figure 6(a). In this case, the formations specify inter-role interactions, namely the positions to which a player should consider passing the ball [25]. Figure 6(b) illustrates the units, the roles involved, and their captains. Here, the units contain defenders, midfielders, forwards, left players, center players, and right players.

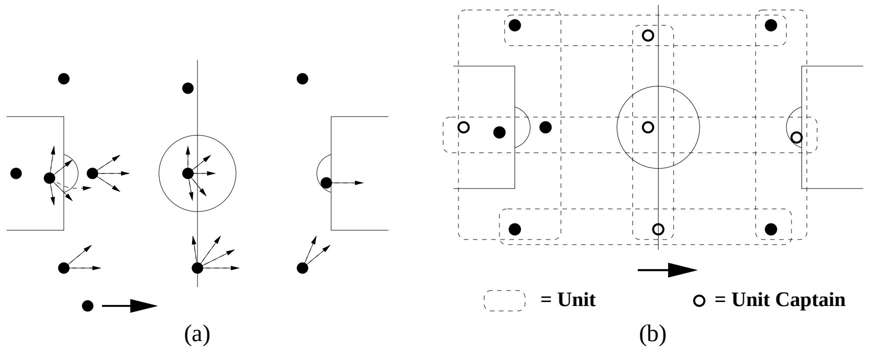

Fig. 6. (a) A possible formation (4-3-3) for a team of 11 players. Arrows represent passing options. (b) Positions can belong to more than one unit.

Since the players are all autonomous, in addition to knowing its own role, each one has its own belief of the team's current formation along with the time at which that formation was adopted, and a map of teammates to positions. Ideally, the players have consistent beliefs as to the team's state, but this condition cannot be guaranteed between synchronization opportunities. Another offshoot of the player's autonomy is that each is free to leave its position unilaterally, leaving the team to adjust behind it. Thus, players are not bound by their positions when presented with unexpected action opportunities.

Our team structure for PTS domains allows for several significant features in our simulated soccer team. These features are: (i) the definition of and switching among multiple formations with units; (ii) flexible position adjustment and position switching; (iii) and pre-defined special purpose plays (set plays).

4.2 Dynamic Switching of Formations

We implemented several different formations, ranging from very defensive (8-2-0) to very offensive (2-4-4).[3] The full definitions of all of the formations are a part of the

[3] Soccer formations are typically described as the X-Y-Z where X, Y, and Z are the number of defenders, midfielders, and forwards respectively. It is assumed that the eleventh player is the goaltender. [11]. Soccer formations are not to be confused with military-type formations in which agents must stay in precise relative positions.

locker-room agreement. Therefore, they are all known to all teammates. However during the periods of full autonomy and low communication, it is not necessarily known what formation the rest of the teammates are using. Two approaches can be taken to address this problem:

- **static formation** - the formation is set by the locker-room agreement and never changes;
- **run-time switch of formation** - during team synchronization opportunities, the team sets globally accessible run-time evaluation metrics as formation-changing indicators.
- **communication-triggered formation switch** - one team member decides that the team should switch formations and communicates the decision to teammates.

The CMUnited-97 simulator RoboCup team used run-time formation switches. Based on the amount of time left relative to the difference in score: the team switched to an offensive formation if it was losing near the end of the game and a defensive formation if it was winning. Since each agent was able to independently keep track of the score and time, the agents were always able to switch formations simultaneously.

Communication-triggered formation switches have also been implemented and tested [22].

4.3 Flexible Positions

In our multi-agent approach, the player positions itself flexibly such that it *anticipates* that it will be useful to the team, either offensively or defensively.

Two ways in which agents can use the position flexibility is to react to the ball's position and to mark opponents. When reacting to the ball's position, the agent moves to a location within its range that minimizes its distance to the ball. When marking opponents, agents move next to a given opponent rather than staying at the default position home. The opponent to mark can be chosen by the player (e.g., the closest opponent), or by the unit captain which can ensure that all opponents are marked, following a preset algorithm as part of the locker-room agreement.

As emphasized throughout, homogeneous agents can play different positions. But such a capability raises the challenging issue of when the players should change positions. In addition, with teammates switching positions, a player's internal player-position map $A \mapsto R$ could become incorrect and/or incomplete. The locker-room agreement provides procedures to the team that allow for coordinated role changing. In our case, the locker-room agreement designates an order of precedence switching among positions within each unit. When a high-priority position is left vacant, players currently filling lower-priority positions consider switching to the recently vacated position. If a player detects that another player is trying to fill the same role, it either vacates the position or informs the other player of the conflict depending on which player is closer to the position's home coordinates.

By switching positions within a formation, the overall joint performance of the team is improved. Position-switching saves player energy and allows them to respond more quickly to the ball.

4.4 Pre-Planned Set Plays

The final implemented improvement facilitated by our flexible teamwork structure is the introduction of set-plays, or pre-defined special purpose plays. As a part of the locker-room agreement, the team can define multi-step multi-agent plans to be executed at appropriate times. Particularly if there are certain situations that occur repeatedly, it makes sense for the team to devise plans for those situations.

In the robotic soccer domain, certain situations occur repeatedly. For example, after every goal, there is a kickoff from the center spot. When the ball goes out of bounds, there is a goal-kick, a corner-kick, or a kick-in. In each of these situations, the referee informs the team of the situations. Thus all the players know to execute the appropriate set-play. Associated with each set-play-role is not only a location, but also a behavior. The player in a given role might pass to the player filling another role, shoot at the goal, or kick the ball to some other location.

For example, Figure 7 illustrates a sample corner-kick set-play. The set-play designates five roles, each with a specific location, which should be filled before the ball is put back into play. Based on the home positions of the current formation, each individual agent can determine the best mapping from positions to set-play locations, i.e. the mapping that requires the least total displacement of the 5 players. If there is no player filling one of the necessary formation roles, then there must be two players filling the same role, one of which must move to the vacant role. In the event that no agent chooses to do so, the set-play can proceed with any single set-play-role unfilled. The only exception is that some player must fill the set-play-role responsible for kicking the ball back into play. A special-purpose protocol is incorporated into the set-play behaviors to guarantee such a condition.

Once the set-play-roles are filled, each player executes the action associated with its set-play-role. As illustrated by the player starting the corner-kick in Figure 7, a player could choose among possible actions, perhaps based on the opponent positions at the time of execution. No individual player is guaranteed of participating in the play. For example, the uppermost set-play position is there just in case one of the other players misses a pass or shoots wide of the goal: no player will pass directly to it. Each player leaves its set-play-role to resume its former role either after successfully kicking the ball, or after a pre-specified, role-specific amount of time.

We found that the set-plays significantly improved our team's performance. During the RoboCup-97 competitions, several goals were scored as a direct result of set-plays.

5 Results

The flexible teamwork structure improves over a rigid structure by way of three characteristics: flexible positioning within roles, set-plays, and changeable formations. We tested the benefits of the first two characteristics by playing a team with flexible, changeable positions and set-plays against a team with rigid positions and no set-plays (default team). The advantage of being able to change formations—the third characteristic—depends on the formation being used by the opponent. Therefore, we tested teams using each defined formation against each other.

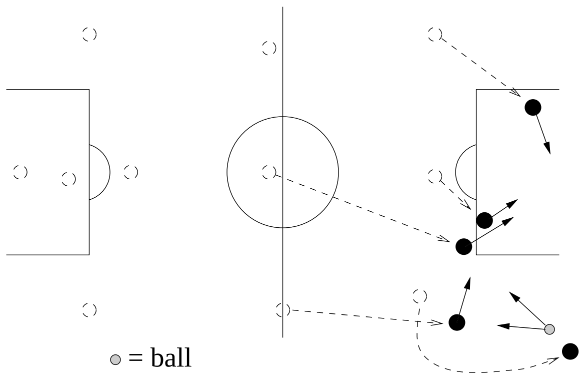

Fig. 7. A sample corner-kick set-play. The dashed circles show the positions in the team's current formation and dashed arrows indicate the set-play-roles—black circles—that they would fill. Solid arrows indicate the direction the ball is to be kicked as part of each set-play-role.

Standard games in the soccer server system last 10 minutes. However, due to the large amount of noise, game results vary greatly. All reported results are cumulative over several games. Compiled statistics include the number of 10-minute games won, the total cumulative goals scored by each team, average goals per game, and the percentage of time that the ball was in each half of the field. The last statistic gives a rough estimate of the degree to which each team was able to control the ball.

5.1 Flexible Positions and Set-Plays

In order to test our flexible teamwork structure, we ran a team using flexible positions with set-plays against one using rigid positions and no set-plays. Both teams used a 4-4-2 formation. As shown in Figure 1, the flexible team significantly outperformed the default team over the course of 38 games.

(Game = 10 min.)	Flexible and Set-Plays	Default
Games won	34	1
Total goals	223	82
Avg. goals	5.87	2.16
Ball in own half	43.8%	56.2%

Table 1. The flexible team won 34 out of 38 games with 3 ties.

Further experimentation showed that both aspects of the flexible team contributed significantly to the team's success. Figure 2 shows the results when a team using flexible positions but no set-plays plays against the default team and when a team using set-plays but rigid positions plays against the default team, again over the course of 38 games. Both characteristics provide a significant advantage over the default team, but they perform even better in combination.

Only Flexible Positions

(Game = 10 min.)	Flexible	Default
Games won	26	6
Total goals	157	87
Avg. goals	4.13	2.29
Ball in own half	44.1%	55.9%

Only Set-Plays

(Game = 10 min.)	Set-Plays	Default
Games won	28	5
Total goals	187	108
Avg. goals	4.92	2.84
Ball in own half	47.6%	52.4%

Table 2. Only using flexible positions and only using set-plays works better than using neither.

5.2 Formations

In addition to the above tests, we tested the various formations against each other, as reported in Table 3. Each entry shows the goals scored for and against when a team using one formation played against a team using another formation over the course of 24 10-minute games. The right-most column collects the total goals scored for and against the team using that formation when playing against all the other teams. In all cases, the teams used flexible positions, but no set-plays.

formations	4-3-3	4-4-2	3-5-2	8-2-0	3-3-4	2-4-4	totals
4-3-3		68-60	68-54	24-28	59-64	70-65	289-271 (51.6%)
4-4-2	60-68		68-46	22-24	51-57	81-50	282-245 (53.5%)
3-5-2	54-68	46-68		13-32	61-72	75-73	249-313 (44.3%)
8-2-0	28-24	24-22	32-13		27-28	45-36	156-96 (61.9%)
3-3-4	64-59	57-51	72-61	28-27		87-69	308-267 (53.6%)
2-4-4	65-70	50-81	73-75	36-45	69-87		293-385 (43.2%)

Table 3. Comparison of our different formations. Entries in the table show the number of goals scored. Total (and percentage) cumulative goals scored against all formations appear in the right-most column.

The results show that the defensive formation (8-2-0) does the best. However the total goals scored when using the defensive formation is quite low. On the other hand, the 3-3-4 formation performs well with a high goal total.

This detailed study allowed us to devise an effective formation-switching strategy for RoboCup-97. Our team [21] used a 4-4-2 formation in general, switching to a 8-2-0 formation if winning near the end of the game, or a 3-3-4 formation if losing. This strategy, along with the flexible teamwork structure as a whole, and the novel communication paradigm, helped us to perform well in the tournament, making it to the semi-finals in a field of 29 teams and out-scoring opponents by a total score of 67-9 [17].

We also used this flexible teamwork structure on our CMUnited-97 small robot team which won the RoboCup-97 small-size robot competition, out-scoring opponents by a total score of 13–1 [17]. Although developed in simulation, all of the teamwork concepts apply directly to real robot teams as well. We were able to reuse the code textually from our simulator clients on the robots and immediately achieve variable formations, flexible positions, and position switching.

Unlike the simulated agents, the robots have a global view, seeing the entire field via an overhead camera. Nonetheless, the agent control modules are distributed, enabling the use of the same team member architecture, including locker-room agreement and formation structure, as was originally developed in simulation. Our robotic system is described in detail in [27].

6 Conclusion

In this paper, we introduced a flexible team structure for periodic team synchronization (PTS) domains. The structure allows for multi-agent tasks using homogeneous agents to be decomposed into flexible roles. Roles are organized into formations, and agents can fill any role in any formation. Agents dynamically change roles and formations in response to changing environments. The team structure includes pre-planning for frequent situations, and agents act individually, but keep the team's goals in mind. This flexible team structure builds upon our team agent architecture, which maintains both an internal and world state, and a set of internal and external behaviors. Coordination is achieved through limited communication and pre-determined procedures as part of a locker-room agreement.

Our teamwork structure will apply in PTS domains such as hospital/factory maintenance, multi-spacecraft missions, search and rescue, and battlefield combat. We presented the implementation of our approach in the robotic soccer domain, which we have used as a substrate to our research. We participated in the RoboCup-97 simulator and small-size real robot competitions. Our flexible team structure approach was developed in the simulator team and subsequently also successfully used in the real robot team.

References

1. Minoru Asada, Eiji Uchibe, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Coordination of multiple behaviors acquired by vision-based reinforcement learning. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS '94)*, pages 917–924, 1994.
2. A. L. C. Bazzan, R. H. Bordini, and J. A. Campbell. Moral sentiments in multi-agent systems. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 113–131. Springer-Verlag, Heidelberg, 1999.

3. Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 41–48, Menlo Park, California, June 1995. AAAI Press.
4. Philip R. Cohen, Hector J. Levesque, and Ira Smith. On team formation. In J. Hintikka and R. Tuomela, editors, *Contemporary Action Theory*. Synthese, 1998. to appear.
5. R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 99–112. Springer-Verlag, Heidelberg, 1999.
6. Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1996.
7. Barbara J. Grosz. Collaborative systems. *AI Magazine*, 17(2):67–85, Summer 1996.
8. L. Hunsberger. Making sharedplans more concise and easier to reason about. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 81–98. Springer-Verlag, Heidelberg, 1999.
9. Jong-Hwan Kim, editor. *Proceedings of the Micro-Robot World Cup Soccer Tournament*, Taejon, Korea, November 1996.
10. Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.
11. Michael L. LaBlanc and Richard Henshaw. *The World Encyclopedia of Soccer*. Visible Ink Press, 1994.
12. A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory, and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.
13. Maja J. Mataric. Interaction and intelligent behavior. MIT EECS PhD Thesis AITR–1495, MIT AI Lab, August 1994.
14. Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multiagent systems: a case study of pass play in soccer. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 63–67, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.
15. J. P. Müller. The right agent (architecture) to do the right thing. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 211–225. Springer-Verlag, Heidelberg, 1999.
16. Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996. Soccer server is available at URL http://ci.etl.go.jp/ noda/soccer/server/index.html.
17. Results of RoboCup-97, 1997. At URL http://www.RoboCup.org.
18. Michael K. Sahota, Alan K. Mackworth, Rod A. Barman, and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.
19. Randy Sargent, Bill Bailey, Carl Witty, and Anne Wright. Dynamic object capture using fast vision tracking. *AI Magazine*, 18(1):65–72, Spring 1997.
20. Peter Stone. Multiagent learning for autonomous spacecraft constellations. In *Proceedings of the NASA Workshop on Planning and Scheduling for Space*, 1997.
21. Peter Stone and Manuela Veloso. The CMUnited-97 simulator team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 387–397. Springer Verlag, Berlin, 1998.

22. Peter Stone and Manuela Veloso. Communication in domains with unreliable, single-channel, low-bandwidth communication. In Alexis Drogoul, Milind Tambe, and Toshio Fukuda, editors, *Collective Robotics*, pages 85–97. Springer Verlag, Berlin, July 1998.
23. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
24. Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48(1):83–104, January 1998.
25. Peter Stone and Manuela Veloso. Using decision tree confidence factors for multiagent control. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 99–111. Springer Verlag, Berlin, 1998.
26. Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:81–124, 1997.
27. Manuela Veloso, Peter Stone, Kwun Han, and Sorin Achim. The CMUnited-97 small-robot team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 242–256. Springer Verlag, Berlin, 1998.

This article was processed using the LaTeX macro package with LLNCS style